django-simple-history Documentation

Release 1.2.3

Corey Bertram

Jun 12, 2017

Contents

django-simple-history stores Django model state on every create/update/delete.

CHAPTER 1

Documentation

Usage

Install

This package is available on PyPI and Crate.io.

Install from PyPI with `pip`:

```
$ pip install django-simple-history
```

Quickstart

To track history for a model, create an instance of `simple_history.models.HistoricalRecords` on the model.

An example for tracking changes on the `Poll` and `Choice` models in the Django tutorial:

```
from django.db import models
from simple_history.models import HistoricalRecords

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    history = HistoricalRecords()

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
    history = HistoricalRecords()
```

Now all changes to `Poll` and `Choice` model instances will be tracked in the database.

Integration with Django Admin

To allow viewing previous model versions on the Django admin site, inherit from the `simple_history.admin.SimpleHistoryAdmin` class when registering your model with the admin site.

This will replace the history object page on the admin site and allow viewing and reverting to previous model versions. Changes made in admin change forms will also accurately note the user who made the change.

An example of admin integration for the `Poll` and `Choice` models:

```
from django.contrib import admin
from simple_history.admin import SimpleHistoryAdmin
from .models import Poll, Choice

admin.site.register(Poll, SimpleHistoryAdmin)
admin.site.register(Choice, SimpleHistoryAdmin)
```

Querying history

Querying history on a model instance

The `HistoricalRecords` object on a model instance can be used in the same way as a model manager:

```
>>> from poll.models import Poll, Choice
>>> from datetime import datetime
>>> poll = Poll.objects.create(question="what's up?", pub_date=datetime.now())
>>>
>>> poll.history.all()
[<HistoricalPoll: Poll object as of 2010-10-25 18:03:29.855689>]
```

Whenever a model instance is saved a new historical record is created:

```
>>> poll.pub_date = datetime(2007, 4, 1, 0, 0)
>>> poll.save()
>>> poll.history.all()
[<HistoricalPoll: Poll object as of 2010-10-25 18:04:13.814128>, <HistoricalPoll:␣
↪Poll object as of 2010-10-25 18:03:29.855689>]
```

Querying history on a model class

Historical records for all instances of a model can be queried by using the `HistoricalRecords` manager on the model class. For example historical records for all `Choice` instances can be queried by using the manager on the `Choice` model class:

```
>>> choice1 = poll.choice_set.create(choice='Not Much', votes=0)
>>> choice2 = poll.choice_set.create(choice='The sky', votes=0)
>>>
>>> Choice.history
<simple_history.manager.HistoryManager object at 0x1cc4290>
>>> Choice.history.all()
[<HistoricalChoice: Choice object as of 2010-10-25 18:05:12.183340>,
↪<HistoricalChoice: Choice object as of 2010-10-25 18:04:59.047351>]
```

Advanced Usage

Version-controlling with South

By default, Historical models live in the same app as the model they track. Historical models are tracked by South in the same way as any other model. Whenever the original model changes, the historical model will change also.

Therefore tracking historical models with South should work automatically.

Locating past model instance

Two extra methods are provided for locating previous models instances on historical record model managers.

as_of

This method will return an instance of the model as it would have existed at the provided date and time.

```
>>> from datetime import datetime
>>> poll.history.as_of(datetime(2010, 10, 25, 18, 4, 0))
<HistoricalPoll: Poll object as of 2010-10-25 18:03:29.855689>
>>> poll.history.as_of(datetime(2010, 10, 25, 18, 5, 0))
<HistoricalPoll: Poll object as of 2010-10-25 18:04:13.814128>
```

most_recent

This method will return the most recent copy of the model available in the model history.

```
>>> from datetime import datetime
>>> poll.history.most_recent()
<HistoricalPoll: Poll object as of 2010-10-25 18:04:13.814128>
```

History for Third-Party Model

To track history for a model you didn't create, use the `simple_history.register` utility. You can use this to track models from third-party apps you don't have control over. Here's an example of using `simple_history.register` to history-track the `User` model from the `django.contrib.auth` app:

```
from simple_history import register
from django.contrib.auth.models import User

register(User)
```

Recording Which User Changed a Model

To denote which user changed a model, assign a `_history_user` attribute on your model.

For example if you have a `changed_by` field on your model that records which user last changed the model, you could create a `_history_user` property referencing the `changed_by` field:

```
from django.db import models
from simple_history.models import HistoricalRecords

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    changed_by = models.ForeignKey('auth.User')
    history = HistoricalRecords()

    @property
    def _history_user(self):
        return self.changed_by

    @_history_user.setter
    def _history_user_setter(self, value):
        self.changed_by = value
```

CHAPTER 2

Code

Code and issue tracker: https://github.com/treyhunner/django-simple-history

Pull requests are welcome.

CHAPTER 3

Changes

1.3.0 (2013-05-17)

- Fixed bug when using `django-simple-history` on nested models package
- Allow history table to be formatted correctly with `django-admin-bootstrap`
- Disallow calling `simple_history.register` twice on the same model
- Added Python 3 support
- Added support for custom user model (Django 1.5+)

1.2.3 (2013-04-22)

- Fixed packaging bug: added admin template files to PyPI package

1.2.1 (2013-04-22)

- Added tests
- Added history view/revert feature in admin interface
- Various fixes and improvements

Oct 22, 2010

- Merged setup.py from Klaas van Schelven - Thanks!

Feb 21, 2010

- Initial project creation, with changes to support ForeignKey relations.